UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/797,238	03/10/2004	Rajeev B. Rajan	MSFT-2924/306986.01	2995

41505 7590 03/24/2010

WOODCOCK WASHBURN LLP (MICROSOFT CORPORATION)
CIRA CENTRE, 12TH FLOOR
2929 ARCH STREET
PHILADELPHIA, PA 19104-2891

EXAMINER	
TIMBLIN, ROBERT M	
ART UNIT	PAPER NUMBER
2167	

MAIL DATE	DELIVERY MODE
03/24/2010	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/797,238	RAJAN ET AL.	
	Examiner	Art Unit	
	ROBERT TIMBLIN	2167	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *12/21/2009*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-5,7-12,14-17,19-22,24 and 30-35* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-5,7-12,14-17,19-22,24 and 30-35* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

This Office Action corresponds to application 10/797,238 filed 3/10/2004.

Response to Amendment

In the present reply (amendments submitted 12/21/2009) Applicant has amended claims 1, 2, 14, 19, and 24. No claims have been added or cancelled. Claims 1-5, 7-12, 14-17, 19-22, 24, and 30-35 are pending.

Claim Objections

Examiner thanks Applicant for the correcting amendments to overcome the prior objection. Accordingly, the previous claim objections have been withdrawn.

However, in light of further examination, Examiner claims 1, 19, and 24 as indicated in the following:

Claim 1 is objected to for unclarity. Specifically, the second to the last limitation on claim page 2 recites "wherein the agent then interacts with the file system to cause execution of file system statement". Therein Examiner recommends inserting "the" or 'a' after "of" and before "file" so to read "execution of the/a file system statement".

Claim 19 is objected to for inconsistent claim language. Specifically, the last limitation on claim page 7 recites "determining if a read lock is available comprises..." and subsequently on the next page recites "if the write lock is not available" and "if the write lock is available…"

Therein, the claim should read "determining if a write lock is available comprises..." as the subsequent limitations refer to the availability of the write lock.

Claim 24 is objected to because "the item" as found in the second limitation lacks antecedent basis.

Appropriate correction is respectfully requested.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 1-5, 7-9, 12, 14-17, 19-22, 24, 30-32, and 35 are rejected under 35 U.S.C. 103(a) as being unpatentable over Cabrera et al. (Cabrera hereafter; U.S. Patent 6,029,160) in view of Stegelmann (U.S. Patent 6,722,155).

With respect to claim 1, Cabrera teaches A method for executing a transaction comprising at least one query language statement and at least one file system statement, each statement relating to a user defined type ("UDT") associated with a database server, the method comprising:

storing at least one field (col. 4 line 62, figure 3; e.g. relation columns) relating to the UDT (col. 4 line 67; e.g. a defined data type) in a relational database table (relation 60, figure 3),

wherein at least one field (col. 4 line 62, figure 3; e.g. relation columns) is a filestream field (col. 5 lines 5-9; e.g. an external file reference (efr) type), and wherein data for each filestream field (col. 5 lines 12-14; e.g. a file reference in the efr) is stored (e.g. reference 17 providing storage for files) in a respective file (fig. 3 reference 70, 74) separate (fig. 4 illustrating a relational database 16 separate from file server 17) from the relational database table (relation 60);

receiving each of the at least one file system statement (col. 5 line 39; e.g. file system calls), wherein each statement (col. 5 line 39; e.g. file system call) comprises a call to open a first item (col. 5 line 39; e.g. call to open a file) and at least one of a call to read from the first item (col. 5 line 40; e.g. call to read a file) and to write to the first item (col. 14 lines 17-18; e.g. calls to rename or delete files), and a call to close the first item (col. 5 line 40; e.g. call to close a file),

receiving each of the at least one query language statement (col. 8 line 29; e.g. SQL query), wherein each query language statement (col. 8 line 29; e.g. SQL query) is associated with a second item (col. 8 line26-46; e.g. SQL performed on a database record);

for each file system statement (col. 5 line 39; e.g. file system call):

upon detecting a storage platform path name (col. 5 line 10, col. 25 line 28; e.g. server/filename) associated with the first item (reference 70, 74; e.g. file) in a file system statement (col. 5 line 39; e.g. file system call) forwarding (fig. 6, step (3)) the file system statement (col. 5 line 39; e.g. file system call) to an agent (reference 45, e.g. file server kernel), wherein the agent (45) performs a call to the storage platform (reference 17; file server and col. 15 line 13) by passing the storage platform path name (col. 5 line 10, col. 25 line 28) to the storage platform (17);

identifying the first item (col. 18 line 36) based upon (col. 10 line 34-36) the storage platform path name (col. 5 line 10, col. 25 line 28; e.g. server/filename);

passing (fig. 6 step (1)) a database engine function (storage component 101) that returns (fig. 6 step (2)) a file system path name (fig. 6; e.g. server/filename) corresponding to the first item (reference 70, 74; e.g. file) to the database server (80, 15, fig. 6);

performing a table look-up (col. 8 line 13; performing a look-up with a database catalog) for the UDT (col. 4 line 67; e.g. a defined data type) associated with the first item (reference 70, 74; e.g. file) on the database server (80, 15, fig. 6);

extracting a real file system path (fig. 6, reference 33) corresponding to the first item (reference 70, 74; e.g. file) using the database engine function;

using the real file system path (fig. 6, reference 33) to perform an operation on the first item (i.e. the path is used to locate in order to perform an operation on reference 70, 74; e.g. file) by passing the real file system path (fig. 6, reference 33) back to the agent (45), wherein the agent (45) then interacts with the file system (17) to cause execution of file system statement (col. 7 line 31-37),

wherein if the file system statement (col. 5 line 39; e.g. file system call) includes open, read and close operations (col. 5 lines 39-40):

creating a transaction (col. 6 lines 27-35 and figs 5-6; e.g. a link operation and user request in a transactional scope) as part of an open operation (col. 5 line 39; e.g. call to open a file), wherein the transaction is managed separately (fig. 1 reference 18 wherein the file manager controls access to files) from the database server (fig. 1, reference 12, 24) and comprises

(wherein the linking operation or request comprises a query to database 16 to access a file in file system 17) the at least one query language statement (col. 8 line 29; e.g. SQL query) and the at least one file system statement (col. 5 line 39; e.g. file system call);

determining whether the at least one query language statement conflicts or the at least one file system statement conflicts with another transaction (col. 20 lines 14-17; e.g. determining if more than one transaction is present);

if the at least one query language statement conflicts or the at least one file system statement conflicts with another transaction, resolving the conflict (col. 20 lines 17-22 wherein previous transactions are resolved);

if the at least one query language statement and the at least one file system statement do not conflict with another transaction (col. 20 line 14; e.g. determining if a transaction count is zero, then no transactions are executing):

obtaining a read lock (col. 14 line 54 wherein a SELECT statement imposes a read lock) on a data table row (relation 60) associated with the associated first item (reference 70, 74; e.g. file) for the file system statement;

performing a read operation in the context of the transaction (col. 5 line 40 and fig. 6; e.g. step (3)); and

committing the transaction as part of a close operation (col. 16 lines 1-6; e.g. committing);

if the file system statement includes open, write and close operations (col. 5 line 40; e.g. call to read a file and col. 14 lines 17-18; e.g. calls to rename or delete):

creating a transaction (col. 6 lines 27-35 and figs 5-6; e.g. a link operation and user request in a transactional scope) as part of an open operation (col. 5 line 39; e.g. call to open a file), wherein the transaction is managed separately (fig. 1 reference 18 wherein the file manager controls access to files) from the database server (fig. 1, reference 12, 24) and comprises (wherein the linking operation or request comprises a query to database 16 to access a file in file system 17) the at least one query language statement (col. 8 line 29; e.g. SQL query) and the at least one file system statement (col. 5 line 39; e.g. file system call);

determining whether the at least one query language statement conflicts or the at least one file system statement conflicts with another transaction (col. 20 lines 14-17; e.g. determining if more than one transaction is present);

if the at least one query language statement conflicts or the at least one file system statement conflicts with another transaction, resolving the conflict (col. 20 lines 17-22 wherein previous transactions are resolved);

if the at least one query language statement and the at least one file system statement do not conflict with another transaction (col. 20 line 14; e.g. determining if a transaction count is zero, then no transactions are executing):

obtaining a write lock (col. 12 line 10 wherein UPDATE operation imposes a write lock) on a data table row (relation 60) associated with the associated first item (reference 70, 74; e.g. file) for the file system statement (col. 5 line 39; e.g. file system call);

performing a write operation in the context of the transaction (col. 15 lines 4-5; e.g. rename/delete call that operate accordingly on a file);

committing the transaction as part of a close operation (col. 16 lines 1-6; e.g. committing);

for each query language statement (col. 8 line 29; e.g. SQL query), starting a transaction (e.g. SELECT/UPDATE) on the database server updating fields (col. 8 line 37) associated with the second item in the query language statement (col. 8 line 29; e.g. SQL query) and sending an updategram to the database server (col. 8 line 66-col. 9 line 3 wherein an update is sent and executed).

Cabrera does not appear to expressly teach while the write lock is obtained, preventing a statement in another transaction or a non-transacted statement from accessing the row; while the write lock is obtained, enabling other statements within the transaction to read from the row.

Stegelmann, however, teaches while the write lock is obtained (col. 17 lines 29-31; e.g. an INSERT/UPDATE/DELETE query results in a pseudo_write lock), preventing a statement in another transaction or a non-transacted statement from accessing the row (col. 17 lines 42-46; e.g. if an update is attempted by a row by transaction T1, that is being modified by another transaction T2, then transaction T1 returns an error);

while the write lock is obtained (col. 17 line 31), enabling other statements within the transaction to read from the row (col. 17 lines 29-35; e.g. after a transaction inserts of updates a row into a table, a subsequent SELECT statement issued by the same transaction will see the inserted row. Therein, the transaction is seen to have insert/update statements as well as a SELECT statement operating on the same row) for providing isolation levels in a transaction.

Accordingly, in the same field of endeavor, (i.e. database integrity), it would have been obvious to one of ordinary skill in the data processing art at the time of the present invention to combine the teachings of the cited references because Stegelmann would have given Cabrera a repeatable read isolation level for the benefit of providing a lower level of locking granularity and ensuring consistency during concurrent transactions.

With respect to claim 2, Cabrera teaches the method of claim 1, wherein the data table row includes a user defined type corresponding to the second item (col. 2 line 45).

With respect to claim 3, Cabrera teaches the method of claim 1, wherein each query language statement is a T-SQL statement (col. 8 line 35-36).

With respect to claim 4, Cabrera teaches the method of claim 1, wherein transactions created for file system statements (col. 5 line 39; e.g. file system calls) are managed (reference 18; file manager) by a storage platform (file system 17).

With respect to claim 5, Cabrera teaches the method of claim 1, further comprising receiving a transaction context for file system operations and performing at least one of a read lock and a write lock consistent with the received transaction context (col. 15 line 45; e.g. a transactional context).

With respect to claim 7, Cabrera teaches the method of claim 1, wherein acquiring the read lock on the row comprises acquiring a read committed view of the row (col. 20 lines 19-22).

With respect to claim 8, Cabrera teaches the method of claim 1, wherein acquiring the write lock on the row comprises acquiring a write lock that will prevent another transaction from accessing the row while the transaction is being processed (col. 20 lines 19-22).

With respect to claim 9, Cabrera teaches the method of claim 1, wherein acquiring the write lock on the row comprises acquiring a write lock that will prevent a non-transacted file system statement from accessing the row while the transaction is being processed (col. 20 lines 19-22 wherein Cabrera implements a lock that prevents access).

With respect to claim 12, Cabrera teaches the method of claim 1, comprising starting the transaction by acquiring one of a read lock (col. 14 line 54 wherein a SELECT statement imposes a read lock) and a write lock (col. 12 line 10 wherein UPDATE operation imposes a write lock) on a filestream field of the row (wherein it is interpreted that if the row is locked, the filestream field within that row is locked as well).

With respect to claim 14, Cabrera teaches A method for locking and isolation of a file system statement, the method comprising:

receiving the file system statement (col. 5 line 39; e.g. file system call) comprising a call to open an item, a call to read from the item, and a call to close the item (col. 5 line 39-40), the file system statement (col. 5 line 39; e.g. file system call) being independent (wherein the file system call is a separate operation) of any database commands employing a query language of a database (col. 8 line26-46; e.g. SQL performed on a database record) and wherein each item (file 70, 74) referenced in the statement (col. 5 line 39; e.g. file system call) is stored separately (fig. 1) from a database table (relation 60) associated with the item (file 70, 74);

upon detecting a storage platform path name (col. 5 line 10, col. 25 line 28; e.g. serveri/filename) associated with the item (file 70, 74) in the file system statement (col. 5 line 39; e.g. file system call), performing a call to a storage platform (reference 17; file server and col. 15 line 13) by passing the storage platform path name (col. 5 line 10, col. 25 line 28; e.g. serveri/filename) to the storage platform col. 5 line 10, col. 25 line 28);

identifying the item (col. 18 line 36) based upon (col. 10 line 34-36) the storage platform path name (col. 5 line 10, col. 25 line 28; e.g. serveri/filename);

passing (fig. 6 step (1)) a database engine function (storage component 101) that returns (fig. 6 step (2)) a file system path name (fig. 6; e.g. server/filename) corresponding to the item (reference 70, 74; e.g. file) to the database server (80, 15, fig. 6);

performing a table look-up (col. 8 line 13; performing a look-up with a database catalog) for the UDT (col. 4 line 67; e.g. a defined data type) associated with the item (reference 70, 74; e.g. file) on the database server (80, 15, fig. 6);

extracting a real file system path (fig. 6, reference 33) corresponding to the item (reference 70, 74; e.g. file) using the database engine function;

using the real file system path (fig. 6, reference 33) to perform an operation on the item (i.e. the path is used to locate in order to perform an operation on reference 70, 74; e.g. file), wherein:

in response to receiving the file system statement (col. 5 line 39; e.g. file system call) that is independent of any database application programming interface requests, determining if a read lock is available for a row of a data table corresponding to the item, wherein determining if a read lock is available comprises determining whether the file system statement conflicts with one of a query language statement and a file system statement associated in a transaction (col. 20 lines 14-17; e.g. determining if more than one transaction is present);

if the read lock is not available for the row of the data table corresponding to the item, then failing the open (col. 15 lines 60 wherein failure of operations is disclosed); and

if the read lock is available for the row of the data table corresponding to the item:

performing a shared open for the item (col. 5 line 39);

acquiring the read lock on the row (col. 14 line 54 wherein a SELECT statement imposes a read lock).

Although Cabrera teaches a data table row associated with a file system statement associated with the transaction (col. 5 lines 36-50, col. 6 lines 27-35), Cabrera does not appear to expressly teach if the statements associated with the transaction do not conflict with another transaction: obtaining a write lock on a data table row associated with a file system statement associated with the transaction; performing a write operation in the context of the transaction; while the write lock is obtained, preventing a statement in another transaction or a non-transacted statement from accessing the row; and while the write lock is obtained, enabling other statements within the transaction to read from the row.

Stegelmann, however, teaches if the statements associated with the transaction do not conflict with another transaction:

obtaining a write lock on a data table row (col. 17 lines 29-31; e.g. an INSERT/UPDATE/DELETE query results in a pseudo_write lock);

performing a write operation in the context of the transaction (col. 17 line 32);

while the write lock is obtained (col. 17 line 31), preventing a statement in another transaction or a non-transacted statement from accessing the row (col. 17 lines 42-46; e.g. if an update is attempted by a row by transaction T1, that is being modified by another transaction T2, then transaction T1 returns an error); and

while the write lock is obtained (col. 17 line 31), enabling other statements within the transaction to read from the row (col. 17 lines 29-35; e.g. after a transaction inserts of updates a row into a table, a subsequent SELECT statement issued by the same transaction will see the

inserted row. Therein, the transaction is seen to have insert/update statements as well as a SELECT statement operating on the same row) for providing isolation levels in a transaction.

Accordingly, in the same field of endeavor, (i.e. database integrity), it would have been obvious to one of ordinary skill in the data processing art at the time of the present invention to combine the teachings of the cited references because Stegelmann would have given Cabrera a repeatable read isolation level for the benefit of providing a lower level of locking granularity and ensuring consistency during concurrent transactions.

With respect to claim 15, Cabrera teaches the method of claim 14, comprising determining if the read lock is available for a row of a data table that includes a user defined type corresponding to the item (col. 2 line 45).

With respect to claim 16, Cabrera teaches the method of claim 14, wherein acquiring the read lock on the row comprises acquiring a read committed view of the row col. 5 line 29-31).

With respect to claim 17, Cabrera teaches the method of claim 14, comprising acquiring the read lock on a filestream field of the row (wherein it is interpreted that if the row is locked, the filestream field within that row is locked as well).

With respect to claim 19, Cabrera teaches A method for locking and isolation of a file system statement, the method comprising:

receiving the file system statement (col. 5 line 39; e.g. file system call) comprising a call to open an item col. 5 line 39-40), a call to write to the item (col. 5 line 40; e.g. call to read a file and col. 14 lines 17-18; e.g. calls to rename or delete), and a call to close the item col. 5 line 39-40), the file system statement (col. 5 line 39; e.g. file system call) being independent (wherein the file system call is a separate operation) of any database commands employing a query language of a database (col. 8 line26-46; e.g. SQL performed on a database record) and wherein each item (file 70, 74) referenced in the statement (col. 5 line 39; e.g. file system call) is stored separately (fig. 1) from a database table (relation 60) associated with the item (file 70, 74);

upon detecting a storage platform path name (col. 5 line 10, col. 25 line 28; e.g. serveri/filename) associated with the item (file 70, 74) in the file system statement (col. 5 line 39; e.g. file system call), performing a call to a storage platform (reference 17; file server and col. 15 line 13) by passing the storage platform path name (col. 5 line 10, col. 25 line 28; e.g. serveri/filename) to the storage platform col. 5 line 10, col. 25 line 28);

identifying the item (col. 18 line 36) based upon (col. 10 line 34-36) the storage platform path name (col. 5 line 10, col. 25 line 28; e.g. serveri/filename);

passing (fig. 6 step (1)) a database engine function (storage component 101) that returns (fig. 6 step (2)) a file system path name (fig. 6; e.g. server/filename) corresponding to the item (reference 70, 74; e.g. file) to the database server (80, 15, fig. 6);

performing a table look-up (col. 8 line 13; performing a look-up with a database catalog) for the UDT (col. 4 line 67; e.g. a defined data type) associated with the item (reference 70, 74; e.g. file) on the database server (80, 15, fig. 6);

extracting a real file system path (fig. 6, reference 33) corresponding to the item (reference 70, 74; e.g. file) using the database engine function;

using the real file system path (fig. 6, reference 33) to perform an operation on the item (i.e. the path is used to locate in order to perform an operation on reference 70, 74; e.g. file), wherein:

in response to receiving the file system statement (col. 5 line 39; e.g. file system call) that is independent of any database application programming interface requests, determining if a read lock is available for a row of a data table corresponding to the item, wherein determining if a read lock is available comprises determining whether the file system statement conflicts with one of a query language statement and a file system statement associated in a transaction col. 20 lines 14-17; e.g. determining if more than one transaction is present);

if the write lock is not available for the row of the data table corresponding to the item, then failing the open (col. 15 lines 60 wherein failure of operations is disclosed); and

if the write lock is available for the row of the data table corresponding to the item:

performing an exclusive open for the item (col. 14 line 54 wherein a UPDATE statement imposes a write lock to impart exclusivity);

acquiring the write lock on the row (i.e. performing the update).

Cabrera does not appear to expressly teach performing a write operation in the context of a transaction; while the write lock is obtained, preventing a statement in another transaction or a non-transacted statement from accessing the row; and while the write lock is obtained, enabling other statements within the transaction to read from the row.

Stegelmann, however, teaches performing a write operation in the context of a transaction (col. 17 line 32);

while the write lock is obtained (col. 17 line 31), preventing a statement in another transaction or a non-transacted statement from accessing the row (col. 17 lines 42-46; e.g. if an update is attempted by a row by transaction T1, that is being modified by another transaction T2, then transaction T1 returns an error); and

while the write lock is obtained (col. 17 line 31), enabling other statements within the transaction to read from the row (col. 17 lines 29-35; e.g. after a transaction inserts of updates a row into a table, a subsequent SELECT statement issued by the same transaction will see the inserted row. Therein, the transaction is seen to have insert/update statements as well as a SELECT statement operating on the same row) for providing isolation levels in a transaction.

Accordingly, in the same field of endeavor, (i.e. database integrity), it would have been obvious to one of ordinary skill in the data processing art at the time of the present invention to combine the teachings of the cited references because Stegelmann would have given Cabrera a repeatable read isolation level for the benefit of providing a lower level of locking granularity and ensuring consistency during concurrent transactions.

With respect to claim 20, Cabrera teaches the method of claim 19, comprising determining if the write lock is available for a row of a data table that includes a user defined type corresponding to the item (col. 2 line 45).

With respect to claim 21, Cabrera teaches the method of claim 19, wherein acquiring the write lock on the row comprises acquiring a write lock that will prevent another statement from accessing the row while the statement is being processed (col. 14 line 54 wherein a UPDATE statement imposes a write lock to impart exclusivity).

With respect to claim 22, Cabrera teaches the method of claim 19, comprising starting the transaction by acquiring the write lock on a filestream field of the row (wherein it is interpreted that if the row is locked, the filestream field within that row is locked as well).

With respect to claim 24, Cabrera teaches A system for executing a file system statement, the system comprising:

a processor (col. 3 line 40; processors);

a relational data engine (DBMS 15) comprising a data table (relation 60) having a row corresponding to the item (reference 83; e.g. the table contains an efr field reference a file stored in a separate file server);

a storage platform (file server 17) built on the relational data engine (DBMS 15 and client application 24), the storage platform (file server 17) comprising means for receiving (44) the file system statement (col. 5 line 39; e.g. file system calls), wherein each item (reference 70, 74; e.g. file) referenced in the statement (col. 5 line 39; e.g. file system call) is stored separately from a database table (relation 60) associated with the item (reference 70, 74; e.g. file);

means for associating (col. 5 line 1-16) the file system statement (col. 5 line 39; e.g. file system call) with a first transaction (col. 6 lines 27-35 and figs 5-6; e.g. a link operation and user request in a transactional scope);

means for associating (col. 5 line 1-16) another file system statement (col. 5 line 40) and a query language statement (col. 8 line 29; e.g. SQL query) with the transaction (col. 6 lines 27-35 and figs 5-6; e.g. a link operation and user request in a transactional scope);

means for determining whether the first transaction conflicts with a second transaction (col. 20 lines 14-17; e.g. TxnCountPtr that determines if more than one transaction is present);

means for resolving the conflict between the first transaction and the second transaction (col. 20 lines 17-22 wherein previous transactions are resolved via means defined in col. 20 line 17-18);

means for starting the transaction (fig. 11, kernel 45) in response to determining that the first transaction does not conflict (col. 20 line 14) with the second transaction and in response to receiving the file system statement by acquiring either a read lock (col. 14 line 54 wherein a SELECT statement imposes a read lock) or a write lock (col. 12 line 10 wherein UPDATE operation imposes a write lock) on the row, the file system statement comprising a call to open a

item (col. 5 line 39; e.g. call to open a file) and at least one of a call to read from the item (col. 5 line 40; e.g. call to read a file) and to write to the item (col. 14 lines 17-18; e.g. calls to rename or delete files), and a call to close the item (col. 5 line 40; e.g. call to close a file), the file system statement being independent (wherein the file system call is a separate operation) of any database commands employing a query language of a database (col. 8 line26-46; e.g. SQL performed on a database record);

a file system (file system 17), wherein the file system is adapted to:

upon detecting a storage platform path name (col. 5 line 10, col. 25 line 28; e.g. serveri/filename) associated with the item (file 70, 74) in the file system statement (col. 5 line 39; e.g. file system call), performing a call to a storage platform (reference 17; file server and col. 15 line 13) by passing the storage platform path name (col. 5 line 10, col. 25 line 28; e.g. serveri/filename) to the storage platform col. 5 line 10, col. 25 line 28);

identifying the item (col. 18 line 36) based upon (col. 10 line 34-36) the storage platform path name (col. 5 line 10, col. 25 line 28; e.g. serveri/filename);

passing (fig. 6 step (1)) a database engine function (storage component 101) that returns (fig. 6 step (2)) a file system path name (fig. 6; e.g. server/filename) corresponding to the item (reference 70, 74; e.g. file) to the item to the relational data engine (DBMS 15);

wherein the relational data engine (DBMS 15) is adapted to:

upon receiving the database engine function (storage component 101) from the file system (17), perform a table look-up (col. 8 line 13; performing a look-up with a database catalog) for an associated user defined type (col. 4 line 67; e.g. a defined data type);

extract a real file system path (fig. 6, reference 33) corresponding to the first item (reference 70, 74; e.g. file) using the database engine function;

pass the real file system path (fig. 6, reference 33) to perform an operation on the item to the file system to cause the file system to perform the operation on the item (col. 7 line 31-37);

wherein the row (relation 60) corresponding to the item (reference 70, 74; e.g. file) includes a user defined type corresponding to the item (col. 2 line 45).

Cabrera does not appear to expressly teach means for preventing a statement in a non-transacted statement or another transaction through a read access or write access from accessing the row while the write lock is obtained; means for enabling other statements within the transaction to read from the row while the write lock is obtained;

Stegelmann, however, teaches means for preventing (col. 17 line 31; e.g. a write lock) a statement in a non-transacted statement or another transaction through a read access or write access from accessing the row while the write lock is obtained (col. 17 lines 42-46; e.g. if an update is attempted by a row by transaction T1, that is being modified by another transaction T2, then transaction T1 returns an error);

means for enabling (col. 17 line 29; e.g. a REPEATABLE READ) other statements within the transaction to read from the row while the write lock is obtained (col. 17 lines 29-35; e.g. after a transaction inserts of updates a row into a table, a subsequent SELECT statement issued by the same transaction will see the inserted row. Therein, the transaction is seen to have insert/update statements as well as a SELECT statement operating on the same row) for providing isolation levels in a transaction.

Accordingly, in the same field of endeavor, (i.e. database integrity), it would have been obvious to one of ordinary skill in the data processing art at the time of the present invention to combine the teachings of the cited references because Stegelmann would have given Cabrera a repeatable read isolation level for the benefit of providing a lower level of locking granularity and ensuring consistency during concurrent transactions.

With respect to claim 30, Cabrera teaches the system of claim 24, wherein the read lock provides a read committed view of the row (col. 20 lines 19-22).

With respect to claim 31, Cabrera teaches the system of claim 24, wherein the write lock prevents another transaction from accessing the row while the transaction is being processed (col. 20 lines 19-22).

With respect to claim 32, Cabrera teaches the system of claim 24, wherein the write lock prevents a non-transacted file system statement from accessing the row while the transaction is being processed (col. 20 lines 19-22 wherein Cabrera implements a lock that prevents access).

With respect to claim 35, Cabrera teaches the system of claim 24, wherein the row comprises a filestream field (relation field 63; e.g. an efr field).

Claims 10, 11, and 33, 34 are rejected under 35 U.S.C. 103(a) as being unpatentable over Cabrera and Stegelmann as applied to parent claims 1 and 24 above, respectively, in view of Anfindsen; U.S. Patent 5,983,225.

With respect to claim 10, although Cabrera teaches acquiring the write lock on the row (col. 12 line 10 wherein UPDATE operation imposes a write lock), Cabrera does not appear to explicitly teach wherein acquiring the write lock comprises acquiring a write lock that will prevent another statement within the transaction from writing to the row.

Anfindsen, however, teaches wherein acquiring the write lock comprises acquiring a write lock that will prevent another statement within the transaction from writing to the row (col. 13 lines 18-19) for determining incompatible (conflicting) transactions.

In the same field of endeavor, (i.e. transaction processing), it would have been obvious to one of ordinary skill in the data processing art at the time of the present invention to combine the teachings of the cited references because the teachings of Anfindsen would have given Cabrera responsive isolation techniques for the benefit of enforcing and providing database integrity (e.g. Cabrera discloses the need and desire to guarantee integrity in col. 2 lines 50-51).

With respect to claim 11, Although Cabrera teaches acquiring the write lock on the row (col. 12 line 10 wherein UPDATE operation imposes a write lock), Cabrera does not explicitly teach wherein acquiring the write lock on the row comprises acquiring a write lock that will enable another statement within the transaction to read from the row.

Anfindsen, however, teaches wherein acquiring the write lock on the row comprises acquiring a write lock that will enable another statement within the transaction to read from the row (col. 13, lines 16-17) for allowing compatible read transactions on a database.

In the same field of endeavor, (i.e. transaction processing), it would have been obvious to one of ordinary skill in the data processing art at the time of the present invention to combine the teachings of the cited references because the teachings of Anfindsen would have given Cabrera responsive isolation techniques for the benefit of enforcing and providing database integrity (e.g. Cabrera discloses the need and desire to guarantee integrity in col. 2 lines 50-51).

With respect to claim 33, Cabrera does not appear to teach the system of claim 24, wherein the write lock prevents another statement within the transaction from writing to the row.

Anfindsen, however, teaches wherein the write lock prevents another statement within the transaction from writing to the row (col. 13 lines 18-19) for determining incompatible (conflicting) transactions.

In the same field of endeavor, (i.e. transaction processing), it would have been obvious to one of ordinary skill in the data processing art at the time of the present invention to combine the teachings of the cited references because the teachings of Anfindsen would have given Cabrera responsive isolation techniques for the benefit of enforcing and providing database integrity (e.g. Cabrera discloses the need and desire to guarantee integrity in col. 2 lines 50-51).

With respect to claim 34, Cabrera does not appear to teach the system of claim 24, wherein the write lock enables another statement within the transaction to read from the row.

Anfindsen, however, teaches wherein the write lock enables another statement within the transaction to read from the row (col. 13, lines 16-17) for allowing compatible read transactions on a database.

In the same field of endeavor, (i.e. transaction processing), it would have been obvious to one of ordinary skill in the data processing art at the time of the present invention to combine the teachings of the cited references because the teachings of Anfindsen would have given Cabrera responsive isolation techniques for the benefit of enforcing and providing database integrity (e.g. Cabrera discloses the need and desire to guarantee integrity in col. 2 lines 50-51).

Response to Arguments

Applicant's arguments with respect to claims 1, 14, 19, and 24 have been considered but are moot in view of the new ground(s) of rejection.

Applicant argues on page 14-15 that Cabrera fails to disclose differentiation between a transaction associated with ah write lock and other transactions or a non-transacted statement. As provided above in a new ground of rejection, Cabrera in view of Stegelmann teach this aspect. For example, Stegelmann teaches while the write lock is obtained (col. 17 lines 29-31; e.g. an INSERT/UPDATE/DELETE query results in a pseudo_write lock), preventing a statement in another transaction or a non-transacted statement from accessing the row (col. 17

lines 42-46; e.g. if an update is attempted by a row by transaction T1, that is being modified by another transaction T2, then transaction T1 returns an error);

while the write lock is obtained (col. 17 line 31), enabling other statements within the transaction to read from the row (col. 17 lines 29-35; e.g. after a transaction inserts of updates a row into a table, a subsequent SELECT statement issued by the same transaction will see the inserted row. Therein, the transaction is seen to have insert/update statements as well as a SELECT statement operating on the same row) for providing isolation levels in a transaction.

Therein, Stegelmann provides a REPEATABLE READ that isolates a transaction with a write lock so that an update attempted by a second transaction is blocked (e.g. results in an error). Thus Stegelmann is respectively interpreted to differentiate at least between a transaction associated with a write lock and other transactions or a non-transacted statement.

Pertinent Art

The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

U.S. Patent Application 2004/0205066 filed by Bhattacharjee et al. The subject matter disclosed therein pertains to the pending claims (i.e. repeatable read).

Conclusion

Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Contact Information

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Robert M. Timblin whose telephone number is 571-272-5627. The examiner can normally be reached on M-Th 8:00-4:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, John R. Cottingham can be reached on 571-272-7079. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/ROBERT TIMBLIN/

Examiner, Art Unit 2167

/John R. Cottingham/

Supervisory Patent Examiner, Art Unit 2167